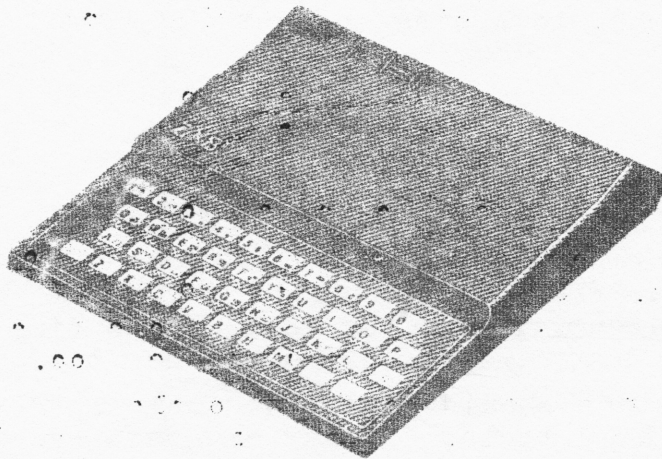


2-1
wy

SINC-LINK



TIMEK-SINCLAIR USERS CLUB

NEWSLETTER

Toronto, Ontario

A MESSAGE FROM THE PRESIDENT

Happy New Year TIMEX SINCLAIR USERS!

This is the first issue of the club's Newsletter under the guidance of our chief contributor and chief machine code expert, Stan Piotrowski. Looks great so far, Stan! Keep up the good work.

Starting the new year with the best intentions, the executive committee plan to bring new ideas and activities to the club. Basic instruction and demonstrations of hardware & software packages that are being used by the members of our club.

As always is the case with all co-operative organizations, I call on all our members to support our efforts and make any suggestions that you feel will improve club activities.

I look forward to seeing and hearing from all our members in 1984.

Greg Lloyd
President

Message From The Newseditor

This is your Newsletter and all contributions will be appreciated. From the brand new programmer to the experienced one...remember that at one time all the more experienced programmers were brand new too.

All submissions should be typed double spaced since I have to retype the article. If any diagrams are included, they must be on a separate page with the proper labels (e.g. fig. 1, etc.).

Regular club members can give the articles to me at the meetings and the out-of-towners can mail it to our club post office box.

In this and the following newsletters, I will have articles for the Basic Programmer and the Machine Code Programmer. Also included will be other little tidbits that you can incorporate into your programs.

Stan Piotrowski
Newseditor

The following are the executive officers of our club and their term of office will be until June 1984.

President: Greg Lloyd
Treasurer: John Roach
Secretary: George Chambers
Librarian: Martin Mauk
News Editor: Stan Piotrowski
Activity Directors: Brian Hammond & Ian Roberts
Meeting Chairman: Harold Goodwin
Out of Town Members: Chris Hart

POOR MAN'S FASTLOAD

by C. Goudeseune

Have you ever noticed what a ZX81 program sounds like near the end? There are 25 bursts of noise with periods of steady tone in between. The reason for this is the manner in which the ROM saves the programs-it copies almost the entire RAM up to the system variable E-LINE. Therefore, the last major item to be saved is the Display File (D-File), consisting of 25 Newlines (the noise) separated by 24 lines of 32 spaces (the steady tone). This is always so because the ZX81 clears the screen before performing any command, including the SAVE.

All these spaces are, of course, completely redundant as far as the program is concerned. A simple way to eliminate them is by adding the following lines to your program:

```
9996 FOR I=1 TO 24
9997   SCROLL
9998   NEXT I
9999   SAVE "program name"
```

To save, type FAST, N/L, RUN 9996, start the tape recorder recording and hit Newline. The reason that the display file seems to disappear is the SCROLL's command operation. When the ROM executes this statement, it clears the bottom line of spaces. This line is truly empty, consisting of only a Newline character without the normal row of spaces. Doing this 24 times empties the whole screen, and the SAVE command in the program executes without first 'clearing' the screen full of spaces again. The FAST statement eliminates the boring wait for the ROM to refill the screen.

This technique is most useful in short programs, as it reduces the loading time by about 10 seconds. In programs approaching 16K in length, 10 seconds makes little difference in 5 or 6 minutes. Of course, this technique will not work with an unexpanded ZX81 or TS1000, because the ROM already performs the screen-emptying as a matter of course to conserve memory. I do not know exactly how QSAVE and the other fastload programs in the library work, and this technique may not be compatible with them.

BASIC PROGRAMMING

Beginning with this article, we will begin using those commands you've learned and apply them to a program. There are many programs in our library that are broad and general and may not be quite suitable to your particular needs and after completing these series of articles, you should be able to customize your own programs. This program is a Data Base program (or D-Base in computer terminology). You have the main program and then you add your own data such as a name & address list or a cheque-book account or a household inventory file. The following program will be a name and address type D-Base. Each and every D-Base program has the same "skeleton" program and you can add the various needed program lines as you wish. Game-type programs are really best left to machine code since it is a lot faster and not so boring.

MENU

After entering many programs I use the following technique. The main Menu always begins at line number 100 and all the routines are located at lines 1000, 2000, etc.

The main menu is the starting point of any program which allows the user to branch off into a choice of operations to be performed. Whenever I start off, I decide what I want in my program and what it should do so lets start off with the menu:

```
100 FAST
110 CLS
120 PRINT AT 3,6;"NAME/ADDRESS FILE"
130 PRINT TAB 6;"17 graphic 7's"
140 PRINT,,TAB 8;"1. ADD"
150 PRINT TAB 8;"2. DELETE"
160 PRINT TAB 8;"3. SORT"
170 PRINT TAB 8;"4. LIST"
180 PRINT TAB 8;"5. SEARCH"
190 PRINT TAB 8;"6. SAVE"
200 PRINT,,TAB 7;"ENTER A NUMBER"
210 SLOW
220 IF INKEY$ < "1" OR INKEY$ > "7" THEN GOTO 220
230 GOTO 1000*VAL INKEY$
```

Explanation: line 140 has 2 commas after the PRINT statement. Remember that 1 comma is a TAB 15 so 2 commas is TAB 31 and the next printed word will be again at TAB 0 resulting in a space inserted between line 130 and 140. This saves typing time and memory. Line 220 is the INKEY\$ function. The computer acts immediately to the keyboard entry. You could have typed INPUT Z\$ but this takes 2 keyboard entries (entering a number then N/L). Therefore, if you type anything else other than a number between 1 and 7 the computer goes back to 220 to wait for the proper response. This is what is known as idiot-proofing the program. Line 230 is what is known as a "computed GOTO". The computer multiplies your response times 1000. Also note we used "VAL". Your response is an INKEY\$ function which is a string function. Since the computed GOTO is a mathematical function your response has to be translated into a real number which is the function of VAL.

ADD Data to File

Before we proceed with this section, there are other program lines we must enter in order for this menu option to work. Enter the following lines and they will all be explained.

NOTE: Line 220 uses firstly, the "less than" sign, then the "greater than" sign.


```

9000 DIM R$(50,88)
9010 DIM N$(26)
9020 DIM A$(23)
9030 DIM T$(20)
9040 DIM C$(7)
9050 DIM P$(12)
9060 DIM W$(80)
9070 DIM S$(26)
9100 LET N=0 (zero not the letter O)
9110 LET L$="-----etc." 32 dashes
9200 GOTO 100

```

Also add: 10 GOTO 9000

In order to be able to add data to a Data Base file, we must first dimension all variables. In line 9000 R\$ will hold all the files or records about to be entered so in the above case we have set aside a space in memory for 50 records containing up to 88 characters. This means you have reserved 4400 bytes of memory ($50 \times 88 = 4400$). The remaining dimensions are for the inputs for entering data so that no entry can exceed the allowed limits. For example, N\$ is dimensioned to 26. If a name is longer than 26 characters, only the first 26 will be entered. The same goes for the others (except W\$ and S\$) If you add them all up your answer will be the allowed characters for R\$ ($26 + 23 + 20 + 7 + 12 = 88$). If possible, always try to have the variables represent their inputs. For example, N\$ is used for Name, C\$ is used for postal Code, etc. W\$ is used for the sorting routine and S\$ is used for the searching routine. N is used for the counter number to keep track of the records currently on file. By letting L\$ = 32 dashes, anytime we need 32 dashes in the program we simply type: PRINT L\$. This will save a lot of memory if used more than once.

NOTE: Don't forget to SAVE your program often since the ZX81 likes to 'crash' a lot.

Now that the DIM of R\$ is explained, you may be wondering why then only 50 records? It is possible to change it to a higher number but it all depends on how much memory you have left after the whole program is entered. A simple test is to change 50 by adding increments of 10 until you get an error message of 4/9000 which means you have run out of memory. Subtract 10 from your trial record number R\$ but remember the more records you have, the longer it takes to save or load. Also, if you have QSAVE then a set amount of memory is reserved above RAMTOP.

For now, add the following lines to the program to ensure your data will not be corrupted. (They will all eventually be changed to their proper line commands.)

```

1000 STOP
2000 STOP
3000 STOP
4000 STOP
5000 STOP
6000 STOP
7000 STOP

```

The following are the 'Add data to record' lines:

```
1000 FAST
1010 CLS
1020 LET N=N+1
1030 IF N > 51 THEN GOTO 9600
1040 PRINT,,,"ADD NEW RECORD TO FILE NO. ";N
1050 PRINT L$
1060 PRINT,,,"NAME: (SURNAME FIRST)"
1065 SLOW
1070 INPUT N$
1080 PRINT AT 6,6;n$
1090 PRINT,,,"ADDRESS: ";
1100 INPUT A$
1110 PRINT A$
1120 PRINT,,,"CITY/TOWN: ";
1130 INPUT T$
1140 PRINT T$
1150 PRINT,,,"POSTAL CODE: ";
1160 INPUT C$
1170 PRINT C$
1180 PRINT,,,"PHONE NUMBER: ";
1190 INPUT P$
1200 PRINT P$
1210 PRINT,,,"TAB 6;"ALL CORRECT (Y/N)"
1220 IF INKEY$="N" THEN GOTO 1300
1230 IF INKEY$="Y" THEN GOTO 1500
1240 GOTO 1220

1300 FAST
1310 CLS
1320 PRINT AT 8,6;"RECORD ";N;" CANCELLED"
1330 PRINT,,,"TAB 9;"PRESS AKEY"
1340 PAUSE 4E4
1350 LET N=N-1
1360 GOTO 1000

1500 FAST
1510 CLS
1520 LET R$(N,1 TO 26)=n$
1530 LET R$(N,27 TO 49)=A$
1540 LET R$(N,50 TO 69)=T$
1550 LET R$(N,70 TO 76)=C$
1570 LET R$(N,77 TO 88)=p$
1580 PRINT AT 8,3;"ADD ANOTHER RECORD (Y/N)"
1590 PAUSE 4E4
1600 IF INKEY$="Y" THEN GOTO 1000
1610 GOTO 100
```

Let's go through the logic in this part of the program. First of all, the record number must be incremented by 1. Since we allowed only 50 records to be entered, it is checked in Line 1030. If the record numbers (N) is less than 51, the program continues; otherwise it will print "FILES FULL" message.

The surname is entered first for searching or sorting routines. Since humans are prone to typing mistakes (like in this Newsletter), Line 1210 checks with you for correctness. If it is not correct, it reduces the record number by 1 then goes back again to entering the data. Notice that correctness checking is done only after all the lines have been entered.

It can be done after each entry or you can add program lines to change any of the entries but remember all those extra lines uses memory and reduces the amount of records you can add.

If everything is correct the program continues at 1500. This is where the inputs are entered into R\$. The characters from 1 to 26 will be R\$(N); if N=1 (your first entry), then N\$ will become R\$(1, 1 to 26). Note that N\$ was dimensioned to 26 characters. After adding a record, if you press BREAK you can check by direct command all or part of R\$. e.g. PRINT R\$(1) will print all of record #1 or PRINT R\$(1, 1 TO 26) will print the name you have entered. A routine later on will actually do this for us.

LIST THE FILE

```
4000 FAST
4010 CLS
4020 LET I=1
4030 PRINT "NO."; TAB 9;"NAME"
4040 PRINT L$
4050 PRINT I;TAB 4; R$(I, 1 TO 26)
4060 LET I=I+1
4070 IF I > N THEN GOTO 4400
4080 IF PEEK 16442 > 4 THEN GOTO 4050
4085 PRINT AT 21,0;"C=CONTINUE, P=PRINTOUT, M=MENU"
4090 PAUSE 4E4
4100 IF INKEY$="P" THEN GOTO 4200
4110 IF INKEY$="C" THEN GOTO 4300
4120 IF INKEY$="M" THEN GOTO 100
4130 GOTO 4090

4200 COPY
4210 GOTO 4090

4300 CLS
4310 GOTO 4030

4400 PRINT AT 20,4;"NO MORE RECORDS ON FILE"
4410 GOTO 4080
```

The above routine will List all the present records on file. The next Newsletter will explain the above routine and continue with the other options available in the menu.

For now, line 4080 will be explained. The system variable address 16442 holds the present line number of the characters printed starting with an empty screen, the address will hold the value 24. (the ZX-81 can print on 24 lines). Everytime a line is filled, the number is decreased by 1. If 20 lines of data have not been yet printed (if the data is available), the program continues until 20 lines are printed and then moves down to line 4085 where the options are printed.

Hopefully, the above program so far will have given you some insight to Basic Programming, the use of various commands, and some knowledge of D-Base programs.

COLLECTION OF TIDBITS

If you find your programs becoming longer and using up too much memory, go through it and anytime you find numbers repeated more than 3 times, then let those numbers=a variable. For example, you may have typed a PRINT TAB 6 (or 8 in the previous program); then let 8=a variable (LET A=6). This serves 2 purposes: a number takes 5 bytes plus each number following. e.g. 567 takes 7 bytes. If repeated 10 times, then that is 70 bytes used whereas A=1 byte only. Another reason is that say TAB 6 is not right. You then have to go through each line changing the 6's to whatever the right TAB number is. By changing the LET statement, you will have then automatically changed all the TAB numbers.

In the slow mode, have you noticed that everytime there is a PAUSE statement, the screen flickers? A much smoother waiting time is a FOR/NEXT loop.

e.g. instead of PAUSE 500 use :

```
XXXX FOR I=1 TO 500      (X's are line numbers)
xxxx NEXT I
```

INPUTS PRINTED IN INVERSE

In some of your programs, you may want all the Inputs to be printed in inverse. For example, Enter the name of your file: and the next line of course will be
XXXX INPUT N\$ (XXXX is your line number)
The following program prints the normal Input and then the inverse of your Input.

```
10 INPUT Y$
20 PRINT AT 10,0;Y$
30 FOR I=1 TO LEN Y$
40 LET X=CODE Y$(I) +128
50 LET Y$(I)=CHR$ X
60 NEXT I
70 PRINT Y$
```

- Line 30 sets up the FOR/NEXT loop the length of your Input. If your Input was "HELLO", then the loop will be 1 TO 5 since "hello" is 5 characters long.
- Line 40 is where the actual inverse takes place. The code of each letter has 128 added to it. If you check in the back of your manual, you will see that the code for "H" is 45. When 128 is added, the new code is 173; looking at the code for 173 is Inverse "H".
- The loop continues until all the inputted characters are inversed then prints it.

MACHINE CODE PROGRAMMING

In some machine code programs you may have disassembled you will have noticed the uses of the OR, AND, XOR, RES, BIT, etc. The uses of these machine code commands are explained in other machine code books so I will only touch on it slightly. In order to use these commands, you must be aware of the binary digits of each code (0 to 255). The last page of this newsletter will have the three most used counting features... decimal, hex, and binary.

Each binary code is made up of 8 digits and are counted from right to left with the first digit being 0. For example, the code for the letter "H" is 45. If you check the binary number for 45, you have: 0010 1101. BIT 0 is 1, etc. to BIT 7 is 0. If you change BIT 7 to a 1, the letter "H" is inversed. Check the new binary digit and you will see that the new binary number is 1010 1101 which is code 173. Checking in your manual, you will find that code 173 is inverse "H". The ZX-81 uses this fact when printing the commands in Basic. Parts of the ROM are tables of the commands and when the subroutine in the ROM goes to print the command word, it checks BIT 7. If it is 0, it continues to print each letter until it finds the seventh BIT to be a 1 and stops printing the command word. For example, the Basic command word "PRINT" will have the first 4 letters in normal mode in the command table but the "T" will be inversed. The routine then stops printing the command word, changes the inverse to normal and returns the keyboard to the user. Therefore, if you are entering a machine code program that will contain tables, this is the fastest and most efficient use of machine code.

I am presently writing a machine code disassembler program that makes use of the above routine. For example, the mnemonic instruction for code 205 is "CALL". The subroutine goes to the word table where 'call' is located and prints the word checking BIT 7 each time until it finds that it is a 1 then stops printing the word. The routine IS long and tedious (like any machine code routine) but it can do all this and fill the screen in less than 1 second since machine code is very, very fast.

Whenever I come across a machine code routine, I may find some parts of the routine interesting or useful in other routines. If you have Toni Bakers's book on "MASTERING MACHINE CODE ON YOUR ZX81", check the games out line by line and you will discover how to use the keyboard scan routine, or routine which I found useful in 2 different programs. One routine decreases a score to zero and another increases a score indefinitely. Take out these routines and check the logic in them and how they are used. The following is how numbers can be increased by one which in machine code is very fast. Another useful feature of this program is how fast machine code cycle is. In about 1 second, it has checked, compared and printed about 1000 numbers. Try that in Basic!!!

First of all, I will give you the program in Basic then the machine code routine.

```
10 LET A=1
20 PRINT AT 10,14;A
30 LET A=A+1
40 GOTO 20
```

Now the machine code routine....

First, enter a REM statement with 35 characters (such as 35 X's. You can enter more if you like but this is a waste of memory).

Now enter the machine code loader program. For those of you don't have a loader program, the following is a machine code loader program:

```
1 REM XXXX....etc (35 X's)
10 LET A=16514
20 INPUT B
30 POKE A,B
40 SCROLL
50 PRINT A;" ";PEEK A
60 LET A=A+1
70 GOTO 20
```

RUN and ENTER the following bytes: (1 at a time)

42,12,64	LD HL, (16396)	:display file location
17,124,1	LD DE, 381	:screen centre
25	ADD HL,DE	
126	CHECK LD A,(HL)	:A contains the character at that location
254,0	CP 0	:is it blank?
32,2	JRNZ +2	:No. It contains a number so jump forward 2 bytes
62,28	LD A,28	:Yes. A now contains the zero digit
60	DEC A	:LET A=A-1
254,38	CP 38	:is A larger than the code for digit "9"?
32,5	JRNZ +5	:No. Jump ahead 5 bytes
54,28	LD (HL),28	:Yes. Print a zero
43	DEC HL	:move display back 1 space
24,239	JR -17	:jump back 17 bytes to CHECK
119	LD (HL),A	:print the digit A contains
205,187,2	CALL 699	:keyboard scan
62,255	LD A,255	
189	CP L	:check for no key pressed
202,130,64	JP Z, 16514	:if a key is pressed, return to Basic & stop.
201	RET	

DECIMAL	HEX	BINARY	DECIMAL	HEX	BINARY
11	0B	10000100	108	06	11000110
12	0C	10000101	109	07	11000111
13	0D	10000110	110	08	11001000
14	0E	10000111	111	09	11001001
15	0F	10001000	112	0A	11001010
16	10	10001001	113	0B	11001011
17	11	10001010	114	0C	11001100
18	12	10001011	115	0D	11001101
19	13	10001100	116	0E	11001110
20	14	10001101	117	0F	11001111
21	15	10001110	118	10	11010000
22	16	10001111	119	11	11010001
23	17	10010000	120	12	11010010
24	18	10010001	121	13	11010011
25	19	10010010	122	14	11010100
26	1A	10010011	123	15	11010101
27	1B	10010100	124	16	11010110
28	1C	10010101	125	17	11010111
29	1D	10010110	126	18	11011000
30	1E	10010111	127	19	11011010
31	1F	10011000	128	1A	11011011
32	20	10011001	129	1B	11011100
33	21	10011010	130	1C	11011101
34	22	10011011	131	1D	11011110
35	23	10011100	132	1E	11011111
36	24	10011101	133	1F	11100000
37	25	10011110	134	20	11100001
38	26	10011111	135	21	11100010
39	27	10100000	136	22	11100011
40	28	10100001	137	23	11100100
41	29	10100010	138	24	11100101
42	2A	10100011	139	25	11100110
43	2B	10100100	140	26	11100111
44	2C	10100101	141	27	11101000
45	2D	10100110	142	28	11101001
46	2E	10100111	143	29	11101010
47	2F	10100100	144	2A	11101011
48	30	10100101	145	2B	11101100
49	31	10100110	146	2C	11101101
50	32	10100111	147	2D	11101110
51	33	10101000	148	2E	11101111
52	34	10101001	149	2F	11110000
53	35	10101010	150	30	11110001
54	36	10101011	151	31	11110010
55	37	10101100	152	32	11110011
56	38	10101101	153	33	11110100
57	39	10101110	154	34	11110101
58	3A	10101111	155	35	11110110
59	3B	10110000	156	36	11110111
60	3C	10110001	157	37	11111000
61	3D	10110010	158	38	11111001
62	3E	10110011	159	39	11111010
63	3F	10110100	160	3A	11111011
64	40	10110101	161	3B	11111100
65	41	10110110	162	3C	11111101
66	42	10110111	163	3D	11111110
67	43	10111000	164	3E	11111111
68	44	10111001			
69	45	10111010			
70	46	10111011			
71	47	10111100			
72	48	10111101			
73	49	10111110			
74	4A	10111111			
75	4B	11000000			
76	4C	11000001			
77	4D	11000010			
78	4E	11000011			
79	4F	11000100			
80	50	11000101			
81	51	11000110			
82	52	11000111			
83	53	11001000			
84	54	11001001			
85	55	11001010			
86	56	11001011			
87	57	11001100			
88	58	11001101			
89	59	11001110			
90	5A	11001111			
91	5B	11010000			
92	5C	11010001			
93	5D	11010010			
94	5E	11010011			
95	5F	11010100			
96	60	11010101			
97	61	11010110			
98	62	11010111			
99	63	11011000			
100	64	11011001			
101	65	11011010			
102	66	11011011			
103	67	11011100			
104	68	11011101			
105	69	11011110			
106	6A	11011111			
107	6B	11100000			
108	6C	11100001			
109	6D	11100010			
110	6E	11100011			
111	6F	11100100			
112	70	11100101			
113	71	11100110			
114	72	11100111			
115	73	11101000			
116	74	11101001			
117	75	11101010			
118	76	11101011			
119	77	11101100			
120	78	11101101			
121	79	11101110			
122	7A	11101111			
123	7B	11110000			
124	7C	11110001			
125	7D	11110010			
126	7E	11110011			
127	7F	11110100			
128	80	11110101			
129	81	11110110			
130	82	11110111			
131	83	11111000			
132	84	11111001			
133	85	11111010			
134	86	11111011			
135	87	11111100			
136	88	11111101			
137	89	11111110			
138	8A	11111111			

Well, that's it fellow club members. Please accept my apologies for any typing errors that may have crept in. If there are errors, try and work out the proper or expected commands. In any case, a correction will be made in the next Newsletter.

Since I have just become the Newseditor, I have tried to have articles for the brand new programmer to the more experienced one. If you have any questions about a programming problem, chances are that others may be having the same problem, so feel free to submit these questions and if possible, an answer will be found to benefit all members.

By now you will have noticed that the format and Newsletter has changed from the previous ones. This is because of a cost savings to the club. Also, since I am new at this type of job, I am learning and with a bit of luck, the Newsletter will get better. I have no idea what type of articles or information you would like to see in your Newsletter so please pass this information to me and I will try to have them in future Newsletters.

Your News Editor,
Stan Piotrowski

Direct all correspondence to one of the Executive officers or:
P.O Box 7274, Stn. A
Toronto, Ont.,
M5W 1X9

